KossLab
KOREA OPEN SOURCE SOFTWARE DEVELOPERS LAB
공개SW개발자 Lab

Titanium 과 Parse Server 를 적용한 고효율의
네이티브 모바일 앱 개발

Titanium_Parse

공개 SW 개발자 Lab 오픈소스프론티어 3 기 김동우

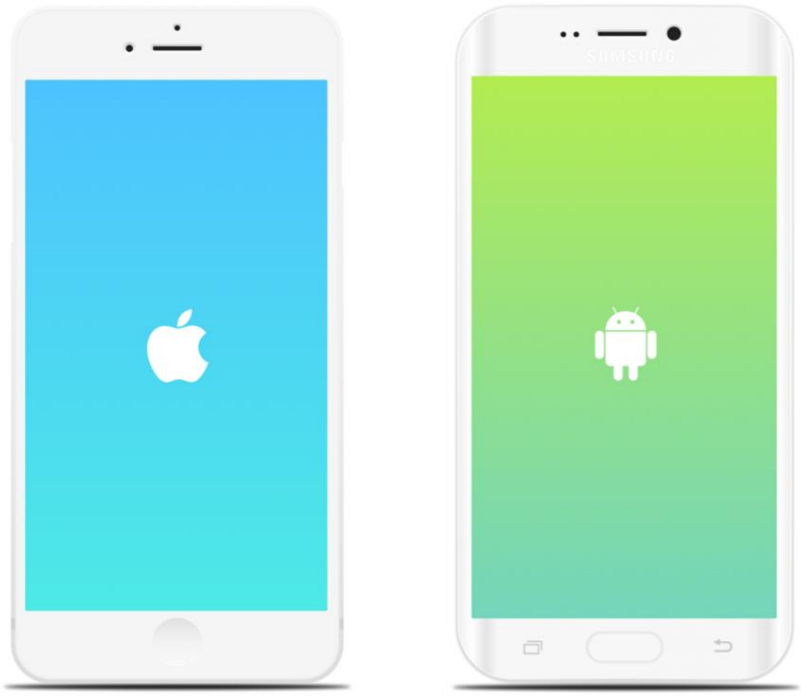
KossLab

전 세계 스마트폰의 O/S는 iOS와 Android가 양분하고 있으며, 하나의 코드로 iOS와 Android 모두를 지원하는 크로스플랫폼 모바일 앱을 개발하는 방법은 끊임없이 발전해왔다.
자바스크립트는 이러한 크로스플랫폼 모바일 앱 개발 요구를 해결할 수 있는 개발 언어로써 많은 프레임워크에 앞다투어 사용되었으며, 오픈소스 커뮤니티의 발전과 함께 폭넓은 인기를 얻고 있다.
스마트폰의 성능 향상, 모바일 O/S의 기술적 성숙으로 사용자의 안목은 네이티브 UI를 갖춘 앱과 이를 웹 브라우저로 흉내 낸 모조품을 구별할 수 있게 되었고, 크로스모바일 앱 개발에서도 네이티브 UI를 제공하는 것이 필요하게 되었다. 이러한 크로스플랫폼 네이티브 모바일 앱 개발을 할 수 있는 대표적인 방법으로 Titanium과 React Native가 있다.
Titanium과 더불어, 오픈소스 MBaaS인 Parse Server를 사용하여 얻을 수 있는 고효율의 네이티브 모바일 앱 개발 방법을 소개한다.

[목차]

1 크로스플랫폼 모바일 앱 개발에서 자바스크립트의 역할

모바일 앱은 iOS 또는 Android 등의 스마트폰 O/S 위에서 구동되는 설치형 애플리케이션을 말한다.
전 세계 스마트폰의 O/S는 iOS와 Android가 양분하고 있으며, 이 구조가 단기간에 급변하거나 한 개의 O/S가 지배적 지위를 가지기 어렵다.



[그림1] 전 세계 스마트폰 O/S 시장을 양분하고 있는 iOS와 Android

그러므로 모바일 앱을 개발하려고 하는 개발자는 iOS와 Android 두 개 버전의 앱을 개발해야 하는 비효율에 빠진다. 만약 한 개의 O/S만을 선택하여 모바일 앱을 개발하게 된다면 상당수의 고객을 포기해야 하는 큰 위험을 안게 된다. 두 O/S는 모바일 앱이 필요로 하는 기능을 구현하기 위해 전혀 호환되지 않은 완전히 다른 코드가 필요하다. iOS는 iOS API를 이용하여 Object C 언어로 개발해야 하며, Android는 Android API를 이용하여 Java 언어로 개발해야 한다. 이 전혀 다른 O/S의 API 및 개발 언어의 학습 비용, 개발 비용은 막대한 부담이 된다. 또한, 개발된 앱의 유지보수 및 기능 업그레이드에 필요한 비용도 지속해서 부담된다.

이러한 비효율을 해결하기 위해서 지속해서 크로스플랫폼 모바일 앱 개발을 위한 기술이 발전해 왔다. 대표적인 두 O/S인 iOS와 Android를 모두 지원하려는 방법으로 표준적 기술인 웹 브라우저를 이용한 방법이 초기에 고안되었으나, 이것으로는 네이티브 UI의 높은 성능과 미려한 디자인을 흉내 내기는 불가능했다.

네이티브 UI를 제공하는 크로스플랫폼 모바일 앱 개발 기술에 대한 요구는 커졌고, 이를 위한 연구는 거듭되었다. 크로스플랫폼 모바일 앱 개발 프레임워크이면서 네이티브 UI를 제공하는 대표적인 제품은 C# 언어를 사용하는 Xamarin과 JavaScript 언어를 사용하는 Titanium, React Native가 있다. JavaScript는 웹 브라우저를 이용한 전통적 방법에서도 로직과 동작을 제어하기 위한 역할을 담당했고, 네이티브 UI를 제공하는 크로스플랫폼 모바일 앱 개발 기술에서도 대표적 두 제품의 개발 언

어로 채택되어 핵심 역할을 담당하고 있다.

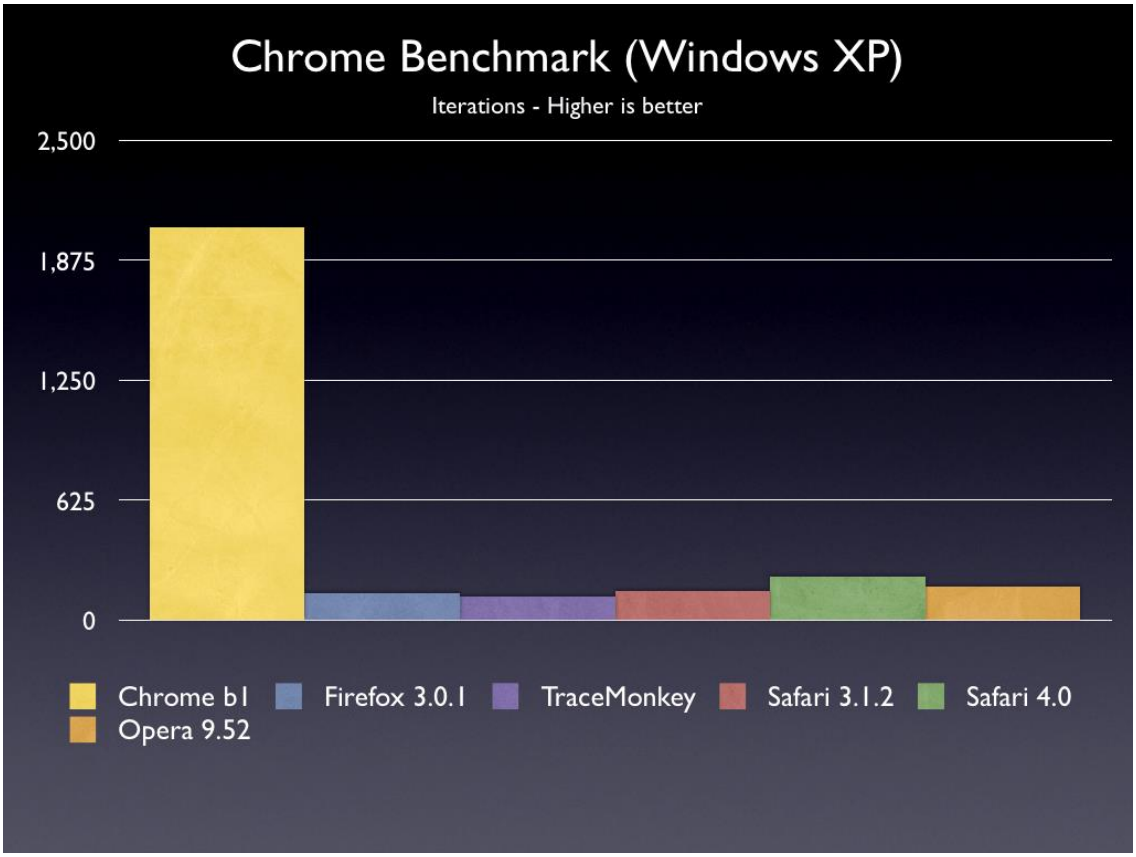
브라우저에 동적인 동작을 구현하기 위해 널리 쓰이던 JavaScript가 크로스플랫폼 네이티브 모바일 앱 개발의 핵심 개발 언어로 사용된 배경에는 이어서 살펴볼 기술 발전의 역사가 존재한다.

2 구글 V8 으로 이룬 자바스크립트의 극적 성능 향상

2008년, 구글은 세상을 놀라게 한 2가지 제품을 발표했다.

첫 번째 제품은 구글이 2008년 9월 2일에 발표한 Chrome 브라우저다. 발표 이후 탄탄한 상승세를 유지한 Chrome은 이제 전통의 강자인 마이크로소프트의 Internet Explorer를 넘어 전 세계 브라우저 점유율의 약 50%를 웃도는 명실공히 최고의 브라우저가 되었다. 이후 Chrome은 Chromium open-source project로 공개되어 브라우저 기술 발달에 크게 기여하고 있다.

두 번째 제품은 구글이 2008년 10월 24일에 발효한 V8 JavaScript Engine이다. V8은 Chrome과 함께 Chromium open-source project의 일부로 JavaScript의 성능향상을 위한 노력의 결정체다.



[그림2] 브라우저 별 자바스크립트 성능 분석 (Google, 2008)

종래의 모든 브라우저를 압도하는 엄청난 성능을 보여준 V8으로 인해 세상은 자바스크립트의 새로운 가능성에 주목하게 된다. 특히 무겁고 반복적인 작업에서 압도적 성능 격차를 보여준 V8으로 인

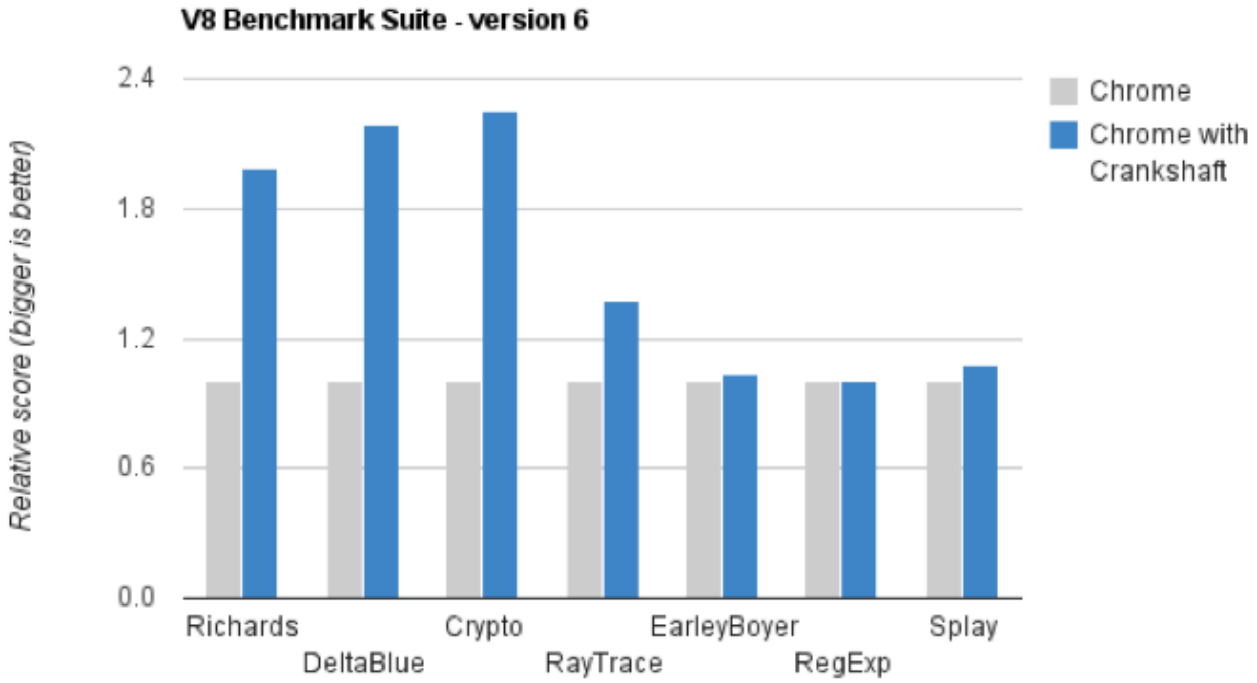
해 이후 브라우저 개발사들은 앞을 다투어 자바스크립트 성능 경쟁에 뛰어들게 된다.

3 네이티브로 도약한 자바스크립트

2009년 5월 27일 Joyent의 개발자 라이언 달(Ryan Dahl)은 구글 V8을 이용한 오픈소스 크로스플랫폼 자바스크립트 런타임 환경인 NodeJS를 발표한다. NodeJS는 이벤트 기반의 논 블로킹 I/O 모델을 사용해 아주 가볍고 효율적이다.

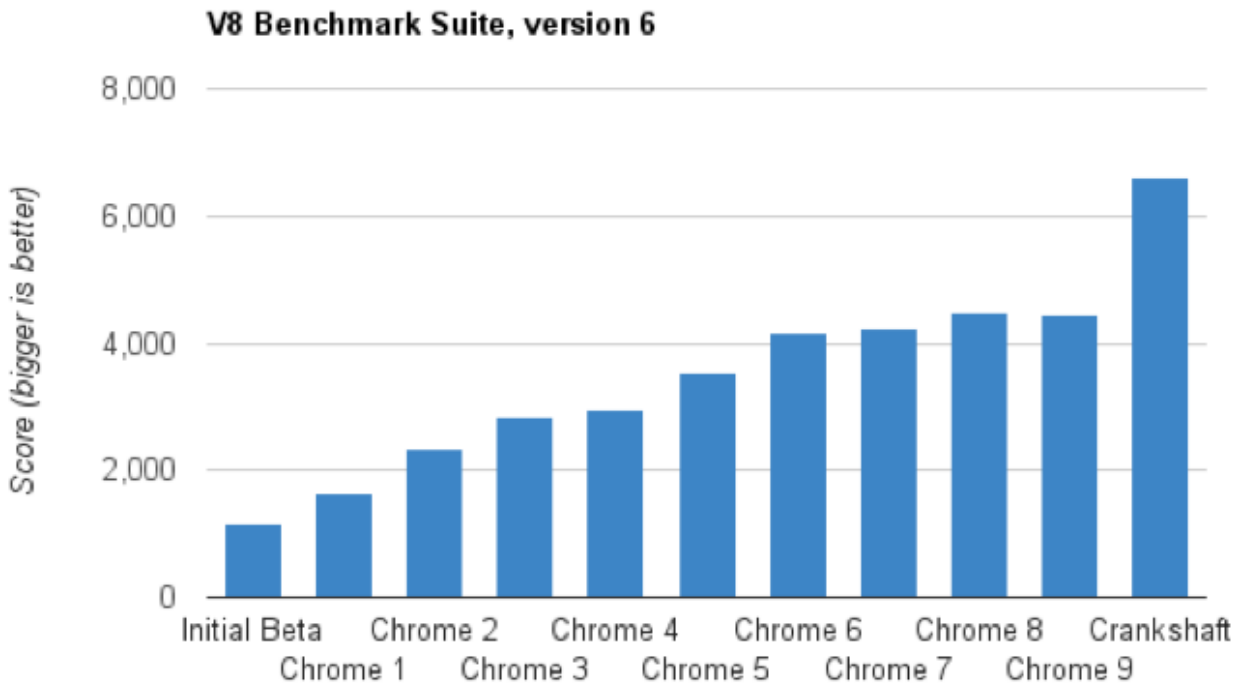
NodeJS의 출시로 자바스크립트는 브라우저를 넘어 네이티브의 세계에 발을 디디게 된다.

이듬해인 2010년 12월 7일, 다시금 구글은 세상을 놀라게 한 V8의 성능 업그레이드를 발표한다. Crankshaft라는 이름의 V8의 새로운 컴파일레이션 인프라스트럭처(compilation infrastructure)는 일거에 종전 V8의 성능을 2배로 끌어올리는 쾌거를 이룩한다.



[그림 3] Crankshaft를 적용한 Chrome의 성능 분석 (Google, 2010)

V8은 자바스크립트를 바이트코드(bytecode)로 컴파일하거나 인터프리트(Interpret)하는 대신 실행하기 전 직접적인 기계어(x86, ARM, MIPS 등)로 컴파일하여 성능을 향상하는 구조로 되어 있었는데, Crankshaft에서는 런타임 프로파일러(runtime profiler)를 이용해 이 컴파일을 필요한 시점에 최적의 효율로 이루어지도록 하여, 성능을 극도로 향상했다.



[그림 4] 2008년 발표 이후 2010년 Crankshaft까지 V8의 성능 비교 (Google, 2010)

V8은 2008년 출시 이후 지속적인 성능향상을 이룩하고 있었으나, Crankshaft로 인해 압도적 성능향상을 보여주었고, 이는 다시 NodeJS에 적용되어 NodeJS의 가능성은 더욱 커지게 된다.

NodeJS는 Crankshaft와 함께, 이에 앞서 2010년 1월 공개되었던 NPM이라는 패키지 매니저와 2011년 6월 발표된 마이크로소프트 윈도우즈용 버전을 통해 네이티브 자바스크립트 런타임으로서의 전성기를 구가하게 되었다.

4 크로스플랫폼 모바일 앱 개발의 완성된 기술, Titanium

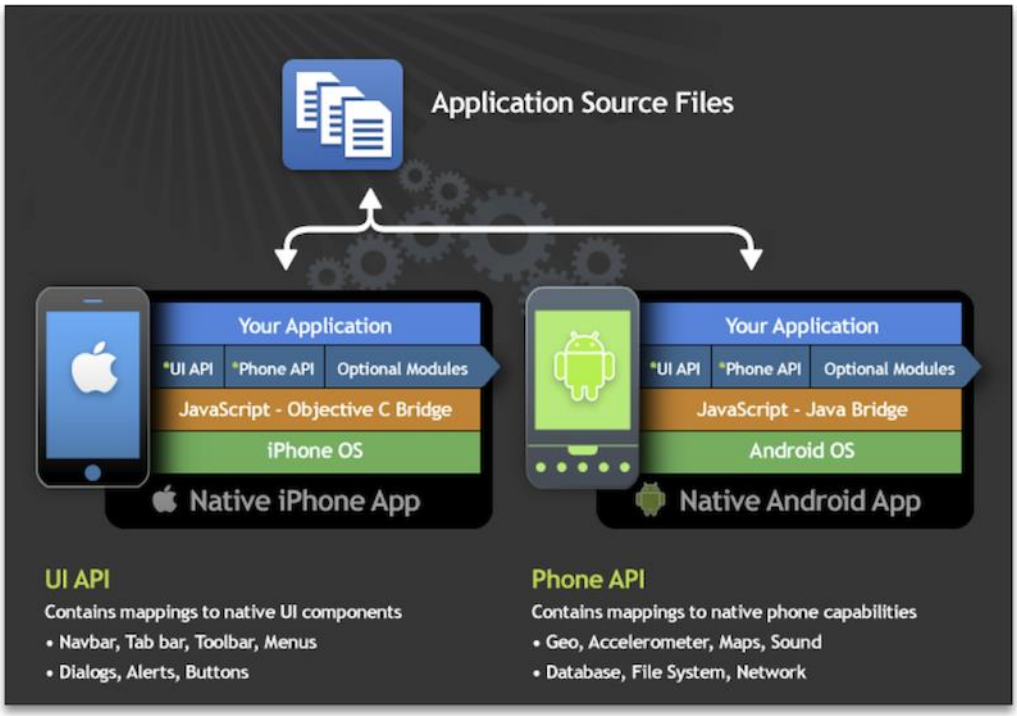
구글이 Chrome과 V8으로 세상을 놀라게 한 2008년, 조용히 세상에 등장한 Titanium이라는 제품이 있다.

Titanium은 오픈소스 프로젝트로 개발된 크로스플랫폼 모바일 앱 개발 프레임워크(cross-platform mobile application development framework)다. 2008년 이후 휘몰아친 자바스크립트 기술의 격동기를 통해, Titanium은 구글의 V8과 NodeJS 등의 오픈소스 자바스크립트 기술을 끊임없이 적용해 급격히 발전하게 된다.

2013년 비즈니스 인사이더(Business Insider)의 발표에 따르면 전 세계 스마트폰의 약 10%에서 Titanium으로 개발된 앱이 구동되었으며, 지속적 발전을 통해 2016년 Titanium 개발자는 전 세계에 80만 명이 넘어서게 된다.

Titanium의 핵심가치는 한 벌의 자바스크립트 코드로 iOS, Android, Windows Phone에서 네이티브

UI로 동작하는 애플리케이션을 만들 수 있다는 엄청난 효율성에 있다.



[그림 5] iOS와 Android에서 네이티브로 동작되는 Titanium의 아키텍처

Titanium의 가장 큰 장점은 7년에 걸쳐 개발된 Titanium SDK가 가진 탄탄한 완성도이다. 오랜 기간 오픈소스로 개발되면서 충분한 시행착오와 기술적 성숙을 거친 Titanium은 모바일 애플리케이션 개발에 필요한 대부분의 API를 기본 제공하고 있으며, 특수한 요구에 필요한 API를 다양한 네이티브 모듈 형태로 조합하여 해결할 수 있도록 제공하고 있다.

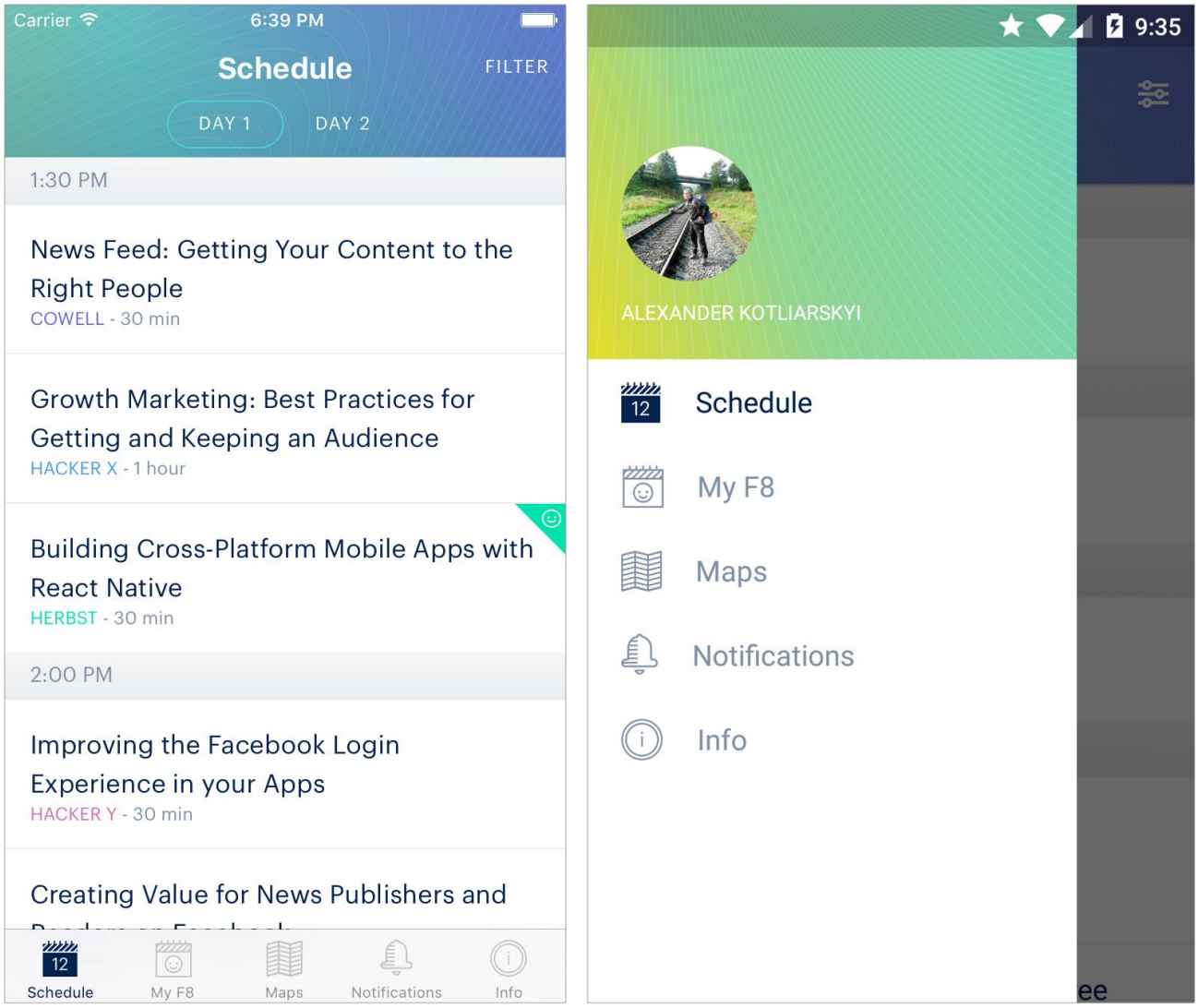
5 iOS 에 이어 Android 를 지원하는 Facebook 의 적자, React Native

앞서 살펴본 Titanium과 유사한 자바스크립트를 이용한 크로스모바일 앱 개발 기술로는 React Native가 있다. React Native는 ReactJS에 기반을 두고 있다. ReactJS는 Facebook, Instagram과 같은 대형 웹 앱에서 UI를 효율적으로 개발하기 위해 Facebook이 자체 개발한 자바스크립트 UI 라이브러리로 2013년 3월 공개되었다. ReactJS는 Facebook의 적극적 지원으로 빠르게 웹 개발 분야에서 높은 인기를 얻었다.

2015년 3월, ReactJS를 통해 축적된 기술을 활용하여 개발된 React Native가 세상에 발표되었다. iOS만을 지원하는 상태로 발표되었던 ReactJS는 2015년 9월 Android를 지원하는 버전으로 업데이트되었다.

하지만 여전히 부족한 기능으로 한계점을 가지고 있던 React Native는 2016년 4월 Facebook의 개

발자 컨퍼런스인 F8의 홍보/안내 목적 모바일 앱이 React Native로 개발되면서, 이제 프로덕션에 사용할 수 있는 수준임을 과시하였다.



[그림 6] React Native로 개발된 'F8 2016' 홍보/안내용 모바일 앱 (Facebook, 2016)

React Native는 Titanium과 구조적으로 매우 유사하면서도, Titanium 및 자바스크립트가 가지는 동기적 동작의 한계점을 보완한 비동기적 구조로 되어 있다. 하지만 여전히 React Native로 iOS/Android를 모두 지원하는 모바일 앱을 개발하기 위해서는 각각의 플랫폼별로 구분해서 작성해야 하는 코드의 양이 상당하며, 플랫폼별 API를 학습해야 하는 약점을 가지고 있다.

하지만 이러한 약점은 Facebook과 오픈소스 개발자들의 적극적 참여와 열광적 인기를 통해 향후 빠르게 개선될 것으로 예견되며, Titanium과 경쟁하며 자바스크립트를 이용한 크로스플랫폼 모바일 앱 개발 분야의 빠른 발전을 가져올 것으로 기대된다.

6 Facebook 이 남긴 MBaaS 의 올인원 솔루션, Parse Server

2011년 Parse라는 미국 캘리포니아 소재의 스타트업이 Parse.com이라는 모바일 백엔드-에즈-어-서비스(mobile backend as a service)를 발표한다. 모바일 앱 개발을 위해 필요한 백엔드 서버를 개발하고 운영하는 데 필요한 데이터 I/O 및 Push 알림 기능 등을 모두 무료 서비스로 이용할 수 있도록 제공함으로써 높은 인기를 얻은 Parse.com은 2013년 8천500만 달러(약 900억 원)에 Facebook에 인수된다.

Parse.com은 Facebook의 지원에 힘입어 빠른 속도로 지원하는 플랫폼을 확장하여, Restful API를 시작으로 iOS / Android / JavaScript / .NET+Xamarin / macOS / Unity / PHP / Arduino / Embeded C 용의 SDK를 오픈소스로 제공하고 있다.

하지만 2016년 1월 28일, Facebook은 돌연 Parse.com 서비스 종료를 예고하며, Parse.com을 리팩토링하여 Parse Server라는 설치형 MBaaS 오픈소스 프로젝트로써 공개한다.

Parse Server는 NodeJS를 기반으로 자바스크립트 언어로 개발되었습니다. 그리고 Parse Server는 데이터베이스로 MongoDB를 사용합니다. MongoDB는 자바스크립트 데이터 형태인 JSON(JavaScript Object Notation)을 그대로 사용하여 높은 개발 효율을 제공하며, 빅데이터를 다룰 수 있는 NoSQL로써 높은 성능, 안정성, 확장성을 제공한다.

자바스크립트 개발능력을 그대로 활용하여 빠르고 효율적으로 Parse Server가 제공하는 기능에 추가로 필요로 하는 기능을 구현해 낼 수 있다.

Parse.com을 사용하고 있던 약 100만여 개의 모바일 앱들은 2017년 1월 28일까지 Parse Server를 이용하여 마이그레이션을 해야 한다는 패닉에 빠졌지만, 오픈소스 개발자 커뮤니티의 힘으로 Parse.com의 제약을 넘어서 놀랄 만큼 빠르게 더 효율적이며 강력한 MBaaS로 거듭나게 되었다.

Facebook은 사업적 손익 계산 때문에 수익률이 떨어지는 Parse.com의 운영종료를 결정한 것으로 추측되지만, 이는 오픈소스 개발자의 집단적 참여를 통해 전화위복이 되어 MBaaS의 가장 인기 있는 솔루션으로 자리 잡았다.

[그림 7] Facebook의 손을 떠나 오픈소스로 거듭난 Parse Server (Parse, 2016)

7 Titanium 과 Parse Server 를 적용한 고효율의 모바일 앱 개발

React Native의 기술 발전의 미래가 매우 기대되기는 하지만, 현재 시점에서 크로스플랫폼 모바일 앱을 개발하기 위해서는 충분히 기술적으로 성숙한 Titanium을 선택하는 것이 안전한 선택이다.

Titanium을 이용해서 Frontend의 높은 효율을 달성했다면, 여기에 더해서 Parse Server를 사용하여 Backend까지도 높은 효율을 달성하여 모바일 앱 개발에 극도의 효율을 추구할 수 있다.

Titanium에 Parse Server를 사용하기 위해서는 Parse Server의 자바스크립트 SDK를 Titanium에 맞게 CommonJS 형식으로 포팅한 모듈을 사용한다. 이 모듈은 역시 오픈소스로 공개되어 있다.

Titanium이 데이터를 다루는 근간은 BackboneJS로 이루어져 있는데, Parse 자바스크립트 SDK 또한 BackboneJS를 바탕으로 개발되어 궁합이 아주 좋다. Titanium의 Frontend 환경에서 개발된 코드를 그대로 Parse Server의 Backend 환경으로 옮겨서 동작할 수 있다. Parse Server 또한 Parse 자바스크립트 SDK를 이용해서 로직을 구성할 수 있도록 제공하기 때문이다.

Titanium과 Parse Server를 조합하여 사용함으로써 Frontend 부터 Backend를 거쳐 데이터베이스까지도 풀스택(Full-stack) 자바스크립트 개발을 할 수 있어서 개발의 효율 및 코드의 수정과 유지보수 모두 아주 높은 효율을 달성할 수 있다. 또한, 자바스크립트로 이루어진 풀스택 개발은 팀을 구성하는 개발자의 수나 구성과 상관없이 커뮤니케이션 비용을 아주 낮게 유지할 수 있다. 참여하는 모든 개발자가 같은 코드 베이스를 이해하고 판단하고 개발할 수 있기 때문이다.

Titanium과 Parse Sever 모두 오픈소스로 개발되기 때문에 폭넓은 자유도와 신뢰성, 오픈소스 커뮤니티의 저력을 바탕으로 한 끊임없는 기술발전을 기대할 수 있다.

Titanium과 Parse Server를 이용하여 크로스플랫폼 모바일 앱을 개발하고, 이를 통해 얻은 높은 효율성은 비즈니스에 더 많은 도전을 할 소중한 기회를 제공할 것이다.